# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/776,328 | 02/02/2001 | Tim Scott Foster | P4756 US | 9582 |

22434 7590 03/10/2005

BEYER WEAVER & THOMAS LLP
P.O. BOX 70250
OAKLAND, CA 94612-0250

| EXAMINER | |
|---|---|
| SHRADER, LAWRENCE J | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2124 | |

DATE MAILED: 03/10/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 09/776,328 | FOSTER ET AL. | |
| | Examiner | Art Unit | |
| | Lawrence Shrader | 2124 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *20 December 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-65* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-65* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

## DETAILED ACTION

1. This office action is in response to the Applicant's RCE and amendments dated 12/20/2004.

2. Claims 1 – 37 remain rejected. New claims 38 – 65 have been added as requested by the Applicant, and they also are rejected. The Applicant's arguments have been fully considered, but are moot in view of the new grounds of rejection.

### *Claim Rejections - 35 USC § 112*

3. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

4. Claims 1, 13, 14, 20, 31, and 38 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

The specification does not disclose: "the test modules do not require the test execution of software components of the software installation package."

See MPEP § 2173.05(i):

Any negative limitation or exclusionary proviso must have basis in the original disclosure. If alternative elements are positively recited in the specification, they may be explicitly excluded in the claims. See *In re Johnson*, 558 F.2d 1008, 1019, 194 USPQ 187, 196 (CCPA 1977) ("[the] specification, having described the whole, necessarily described the part remaining."). See also *Ex parte Grasselli*, 231 USPQ 393 (Bd. App. 1983), *aff'd mem.*, 738 F.2d 453 (Fed. Cir. 1984). The mere absence of a positive recitation is not basis for an exclusion.

Any claim containing a negative limitation, which does not have basis in the original disclosure, should be rejected under **35 U.S.C. 112**, first paragraph, as failing to comply with the written description requirement. Note that a lack of literal basis in the specification for a negative limitation may not be sufficient to establish a *prima facie* case for lack of descriptive support. *Ex parte Parks*, 30 USPQ2d 1234, 1236 (Bd. Pat. App. & Inter. 1993). See **MPEP § 2163 - § 2163.07(b)** for a discussion of the written description requirement of **35 U.S.C. 112**, first paragraph.

### *Claim Rejections - 35 USC § 101*

5. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

6. Claim 13 is rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. "A computer program on a *carrier* medium" is non-statutory because the specification states that the carrier medium could be a transmission medium, which could be a non-physical medium. It is suggested that the claim be amended to say a "storage medium."

### *Claim Rejections - 35 USC § 103*

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1, 2, 8, 11, 12, 44 – 46, 47 – 51; 13; 14, 15, 19, 55 – 59; 20, 21, 60 – 62; 22; 23, 24, 28, 63 – 65; 31; and 38 – 43 are rejected under 35 U.S.C. 103(a) as being unpatentable over

Logan, U.S. Patent 6,601,018 in view of Breggin et al., U.S. Patent 6,560,776 (hereinafter referred to as Breggin).

**In regard to claim 1:**

> *"a framework operable to identify at least one test module defining a test of at least one parameter of the at least one software component of the package, wherein the test modules do not require the test execution of software components of the software installation package;"*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components (column 2, lines 19 – 29), but does not disclose that the test modules do not require the execution of the software. However. Breggin discloses that software verification is conducted without execution of the software package (e.g., Figures 1 – 3b; column 9, lines 9 – 37). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the test skeleton as taught by Logan with the verification tool as taught by Breggin, because the combination allows the user to discover differences as taught by Breggin at column 11, lines 5 – 8, and make corrections before the software is executed.

> *"a control module operable to access the framework to cause the at least one test module identified therein to perform the test defined thereby for verifying the package."*

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module, which accesses the client framework (Logan column 4, lines 40 – 42; column 31 – 39) to perform a test.

**In regard to claim 2, incorporating the rejection of claim 1:**

> *"...wherein the framework identifies a plurality of test modules."*

Logan discloses a test suite in an automatic test framework (column 2, lines 19 – 50).

**In regard to claim 8, incorporating the rejection of claim 2:**

> *"...wherein the framework comprises a directory having a plurality of entries, each entry identifying one of the plurality of test modules."*

Logan discloses a repository for a plurality of test suites allowing the tester to know where to find (a directory) a specific test suite (column 7, lines 3 – 25).

**In regard to claim 11, incorporating the rejection of claim 2:**

> *"...wherein each of the plurality of test modules is formed by a script and the framework identifies each of the test modules by a name for the script."*

Logan teaches the plurality of test modules formed by a script, which is called by name (see column 5 code example).

**In regard to claim 12, incorporating the rejection of claim 2:**

> *"...wherein each of the test modules is formed by a software object."*

A test case (module) is formed by a software object as shown in column 5 of Logan.

**In regard to claim 44, incorporating the rejection of claim 2:**

> *"...wherein the plurality of test modules of the framework define a series of automatically executable tests and wherein the test modules do not require the test execution of software components of the software installation package."*

Logan discloses a repository of test suites in an automatic test framework (column 2, lines 19 – 50; column 7, lines 3 – 8), but does not disclose that the test modules do not require the execution of the software. However. Breggin discloses that software verification is conducted without execution of the software package (e.g., Figures 1 – 3b; column 9, lines 9 – 37). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the test skeleton as taught by Logan with the verification tool as taught by

Breggin, because the combination allows the user to discover differences as taught by Breggin at column 11, lines 5 – 8, and make corrections before the software is executed.

**In regard to claims 45 and 47, incorporating the rejection of claim 44:**

> *"...wherein the software installation package comprises a file list that identifies a plurality of software components included in the software package and includes data about the software components of the software package; and*
>
> *wherein framework identifies a plurality of test modules."*
>
> *"...wherein the plurality of test modules are operable to use the data included in the file list to verify the software package."*

Logan discloses a repository of test suites in an automatic test framework (column 2, lines 19 – 50; column 7, lines 3 – 8).

**In regard to claim 48, incorporating the rejection of claim 47:**

> *"...wherein the data included in the file list includes parameter information concerning the plurality of software components in the software installation package; and*
>
> *wherein the plurality of test modules are operable to use the parameter information included in the file list to verify the software package."*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist of parameters (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61).

**In regard to claim 49, incorporating the rejection of claim 48:**

> *"...wherein the parameter information in the file list includes at least one of: file names for the software components, version numbers for the software components, vendor identification for the software components, copyright information concerning the software components, the size of the software components, the binary data types of the software components."*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist for copyright information (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61).

**In regard to claim 50, incorporating the rejection of claim 48:**

> *"...wherein the parameter information in the file list includes the compiler version used with the software components."*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist of parameters (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61), but does not explicitly disclose compiler version. However, one skilled in the art would have known that a plurality of parameters could be chosen depending on the type of test, and it would be reasonable to expect that compiler version would be checked. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the framework generating a test skeleton as discloses by Logan with the well known ability to check a compiler version because one would be motivated to verify the version level of a component based on the compiler used to produce the code under test.

**In regard to claim 51, incorporating the rejection of claim 48:**

> *"...wherein the parameter information in the file list includes at least one of: copyright information concerning the software installation components, the size of the software components, the binary data types of the software components."*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist for copyright information (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61).

**In regard to claims 52 and 53, incorporating the rejection of claim 2:**

*"...wherein test modules of the framework can be added, deleted, or modified creating a flexible framework."*

*"...wherein test modules can be added and deleted from the framework and can be modified to provide a flexible framework."*

See Logan abstract where test suites are generated for individualized test cases of a component.

**In regard to claim 13:**

*"a) forming a framework operable to identify at least one test module defining a test of at least one parameter of the at least one software component of the package, wherein the test module does not require the test execution of a software component of the software package;"*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61), but does not disclose that the test modules do not require the execution of the software. However. Breggin discloses that software verification is conducted without execution of the software package (e.g., Figures 1 – 3b; column 9, lines 9 – 37). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the test skeleton as taught by Logan with the verification tool as taught by Breggin, because the combination allows the user to discover differences as taught by Breggin at column 11, lines 5 – 8, and make corrections before the software is executed.

*"b) forming a control module operable to access the framework to cause the at least one test module identified therein to perform the test for verifying the package."*

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module, which accesses the client framework (column 4, lines 40 – 42; column 31 – 39).

**In regard to claim 14:**

> *"a) providing a framework for identifying at least one test module, each said test module defining a test of at least one parameter of the at least one software component of the package wherein the test is configured to use the data entries of the file list to test the at least one parameter of the software package and wherein the testing is accomplished without performing a test execution of a software component of the software installation program;"*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist of parameters (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61), but does not disclose that the test modules do not require the execution of the software. However. Breggin discloses that software verification is conducted without execution of the software package (e.g., Figures 1 – 3b; column 9, lines 9 – 37). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the test skeleton as taught by Logan with the verification tool as taught by Breggin, because the combination allows the user to discover differences as taught by Breggin at column 11, lines 5 – 8, and make corrections before the software is executed.

> *"b) accessing the framework to identify the at least one test module;"*

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module, which accesses the client framework (column 4, lines 40 – 42; column 31 – 39).

*"c) causing the at least one test module to perform the test defined thereby on the package."*

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module causing the initiation of a test on the client framework (column 4, lines 40 – 42; column 31 – 39).

**In regard to claim 15, incorporating the rejection of claim 14:**

*"...wherein the framework identifies a plurality of the test modules."*

Logan discloses a test suite in an automatic test framework (column 2, lines 19 50).

**In regard to claim 19, incorporating the rejection of claim 15:**

*"...providing a directory in the framework, wherein the directory has a plurality of entries, each entry identifying one of the plurality of test modules."*

Logan discloses a repository for a plurality of test suites allowing the tester to know where to find (a directory) a specific test suite (column 7, lines 3 – 25).

**In regard to claim 55,** incorporating the rejection of claim 54:

*"...wherein c) causing the at least one test module to perform the test defined thereby on the package includes performing the test using parameters included in the file list to verify the software package."*

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module causing the initiation of a test on the client framework (column 4, lines 40 – 42; column 31 – 39).

**In regard to claim 56,** incorporating the rejection of claim 55:

*"...wherein the parameter information in the file list includes at least one of: file names for the software components, version numbers for the software components, vendor identification for the software components, copyright information concerning the software components, the size of the software components, the binary data types of the software components."*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist for copyright information (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61).

**In regard to claim 57,** incorporating the rejection of claim 55:

> *"...wherein the parameter information in the file list includes the compiler version used with the software components."*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist of parameters (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61), but does not explicitly disclose compiler version. However, one skilled in the art would have known that a plurality of parameters could be chosen depending on the type of test, and it would be reasonable to expect that compiler version would be checked. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the framework generating a test skeleton as discloses by Logan with the well known ability to check a compiler version because one would be motivated to verify the version level of a component based on the compiler used to produce the code under test.

**In regard to claim 58**, incorporating the rejection of claim 55:

> *"...wherein the parameter information in the file list includes at least one of: copyright information concerning the software installation components, the size of the software components, the binary data types of the software components."*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist for copyright information (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61).

**In regard to claim 59,** incorporating the rejection of claim 55:

> *"...wherein the providing the framework includes added and deleted test modules from the framework as desired."*

See Logan abstract where test suites are generated for individualized test cases of a component.

**In regard to claim 20:**

> *"a) "at last one test module configured to use the data entries of the file list to test at least one parameter of the software installation package without performing a test execution of a software component of the software installation package;"*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist of parameters (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61), but does not disclose that the test modules do not require the execution of the software. However. Breggin discloses that software verification is conducted without execution of the software package (e.g., Figures 1 – 3b; column 9, lines 9 – 37). Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the test skeleton as taught by Logan with the verification tool as taught by Breggin, because the combination allows the user to discover differences as taught by Breggin at column 11, lines 5 – 8, and make corrections before the software is executed.

> *"b) a framework to identify at least one test module;"*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components (column 2, lines 19 – 29).

> *"c) a control module operable to access the framework for causing the at least one test module identified therein to perform the test defined thereby for verifying the package."*

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module, which accesses the client framework (column 4, lines 40 – 42; column 31 – 39).

**In regard to claim 21,** incorporating the rejection of claim 20:

> *"...wherein the system comprises a computer including a processor, memory and software held in the memory and operable to control the processor, the software forming: said framework and said control module."*

Logan discloses a framework operating on computer system having a processor, memory, and software held n memory (column 3, lines 28 – 64).

**In regard to claim 60,** incorporating the rejection of claim 21:

> *"...wherein the test modules are configured to verify the software installation package by testing the software components of the software installation package using data entries of the file list to test parameters including at least one of: file names for the software components, version numbers for the software components, vendor identification for the software components, copyright information concerning the software components, the size of the software components, the binary data types of the software components."*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist for copyright information (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61).

**In regard to claim 61,** incorporating the rejection of claim 21:

> *"...wherein the test modules are configured to verify the software installation package by testing data entries of the file list to determine if the correct compiler version is used with the software components."*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist of parameters (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61), but does not explicitly disclose compiler version. However, one skilled in

the art would have known that a plurality of parameters could be chosen depending on the type of test, and it would be reasonable to expect that compiler version would be checked. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the framework generating a test skeleton as discloses by Logan with the well known ability to check a compiler version because one would be motivated to verify the version level of a component based on the compiler used to produce the code under test.

**In regard to claim 62,** incorporating the rejection of claim 21:

> *"...wherein the test modules are configured to verify the software installation package by testing data entries of the file list to determine if the parameter information in the tile list includes correct information concerning at least one of: copyright information concerning the software installation components, the size of the software components, the binary data types of the software components."*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist for copyright information (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61).

**In regard to claim 22:**

> *"a) a memory for storing software;"*

Logan discloses a framework operating on computer system having a processor, memory, and software held in memory (column 3, lines 28 – 64).

> *"b) a processing unit for executing the software to carry out the steps of*
> *(i) providing a framework to identify at least one test module defining a test of at least one parameter of the at least one software component of the package; and*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components (column 2, lines 19 – 29).

> *(ii) providing a control module operable to access the framework for causing the at least one test module identified therein to perform a test that uses the data entries of the file list to test the at least one parameter of the software package thereby verifying the package wherein the file list data entries are associated with parameters concerning at least one of compiler versions used with the software components, copyright information concerning the software components, the size of the software components, the binary data types of the software components."*

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module, which accesses the client framework (column 4, lines 40 – 42; column 31 – 39), and discloses a checklist for copyright information (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61).

**In regard to claim 23:**

> *"a) providing a framework for identifying at least one test module, each said test module configured to use the data entries of the file list to test at least one parameter of the software package thereby defining a test of at least of at least one parameter of the at least one software component of the package;"*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist of parameters (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61).

> *"b) accessing the framework to identify the at least one test module;"*

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module, which accesses the client framework (column 4, lines 40 – 42; column 31 – 39).

*"c) causing the at least one test module to perform the test defined thereby on the package."*

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module causing the initiation of a test on the client framework (column 4, lines 40 – 42; column 31 – 39).

**In regard to claim 24,** incorporating the rejection of claim 23:

*"...wherein the framework identifies a plurality of test modules."*

Logan discloses a test suite in an automatic test framework (column 2, lines 19 50).

**In regard to claim 28**, incorporating the rejection of claim 24:

*"...providing a directory in the framework, wherein the directory has a plurality of entries, each entry identifying one of the test modules."*

Logan discloses a repository for a plurality of test suites allowing the tester to know where to find (a directory) a specific test suite (column 7, lines 3 – 25).

**In regard to claim 63**, incorporating the rejection of claim 23:

*"...wherein the plurality of test modules are configured to test at least one of: that the software components are compiled using the same compiler version; that binary data types of the software components are compatible with the same architecture; that the copyrights are current;, and a test whether only specified software components are present in the installation package."*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist for copyright information (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61).

**In regard to claim 64**, incorporating the rejection of claim 63:

*"...wherein the plurality of test modules are further configured to include a module for testing whether there have been changes to the software installation package relative to a prior version of the software installation package."*

Logan checks for installation software version (column 7, lines 18 – 22).

**In regard to claim 65**, incorporating the rejection of claim 63:

> *"...wherein the plurality of test modules are further configured to include a module for testing whether there are any zero size files in the software installation package."*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components (column 2, lines 19 – 29), but does not explicitly disclose whether there are any zero size files in the software installation package. However, one skilled in the art would have known to check for a zero file size as a condition for defining the test modules. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the framework generating a test skeleton that tests software components as disclosed by Logan combined with the well known procedure to check for a zero file size before running a test in order to have a something available to test.

**In regard to claim 31:**

> *"a) receiving the software package wherein the software package includes a file list having data entries associated with parameters for the at least one software component;"*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist of parameters (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61).

> *"b) accessing a framework that references at least one test module to identify the at least one test module from the framework, each said test module configured to use the data entries of the file list to define a test of the software package;"*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components with a checklist for copyright information (column 2, lines 19 – 29; column 5, line 66 to column 6, line 61).

*"c) performing the test defined by the at least one test module on the package."*

A web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module, which accesses the client framework (column 4, lines 40 – 42; column 31 – 39) to perform a test.

**In regard to claim 38:**

*"a) providing a test framework that includes a control module and at least one test module suitable for conducting verification of software installation packages;*
*b) receiving the software installation package;*
*c) executing an initial verification of the software installation package using a control module, wherein the initial verification is conducted without performing a test execution of the software components of the software installation package;*
*d) where the initial verification is successful, selecting a next module from among the at least one test module to conduct verification of the software package; and*
*e) executing verification testing of the software package using the next module wherein the verification using the next module is conducted without performing a test execution of the software components of the software installation package."*

Logan discloses a framework generating a test skeleton (module defining a test) that tests software components (column 2, lines 19 – 29), a web browser interfaced to an integrated test environment (e.g., JAVA Development Kit) acts as a control module, which accesses the client framework (Logan column 4, lines 40 – 42; column 31 – 39) to perform a test, but does not disclose that the test modules do not require the execution of the software. However. Breggin discloses that software verification is conducted without execution of the software package (e.g., Figures 1 – 3b; column 9, lines 9 – 37). Therefore, it would have been obvious to one skilled in

the art at the time the invention was made to combine the test skeleton as taught by Logan with the verification tool as taught by Breggin, because the combination allows the user to discover differences as taught by Breggin at column 11, lines 5 – 8, and make corrections before the software is executed.

**In regard to claim 39**, incorporating the rejection of claim 38:

> *"... wherein operations d and e are iteratively performed until all test modules of the at least one test module have been executed."*
>
> See Logan at column 7, lines 4 – 9)

**In regard to claim 40**, incorporating the rejection of claim 38:

> *"...wherein if any verification operations fail an error message is generated."*
>
> Logan handles errors at column 8, lines

**In regard to claim 41**, incorporating the rejection of claim 38:

> *"...wherein executing the initial verification of the software installation package comprises:*
>
> *receiving a verification command; and*
>
> *checking that the verification command includes a correct number of arguments."*
>
> Logan discloses test results (column 11, lines 21 – 40).

**In regard to claim 42**, incorporating the rejection of claim 41:

> *"...wherein executing the initial verification of the software installation package comprises at least one of:*
>
> *confirming that a user of the software installation package has the correct permissions;*
>
> *confirming that software components of the software installation package exist in the correct directories;*
>
> *confirming that the software components do not include any zero size files;*

*confirming that the software components comprise actual data files and not data links; and*

*confirming that package map and package information files for the software package exist in a Solaris, environment;"*

Logan discloses utilities to perform a plurality of functions to maintain framework efficiency (column 11, lines 41 – 48).

**In regard to claim 43**, incorporating the rejection of claim 38:

*"...wherein a) providing a test framework that includes a control module and at least one test module suitable for conducting verification of software installation packages includes adding, deleting, and modifying the at least one test module to provide a flexible test framework."*

See Logan column 11, lines 21 – 40 for repetition of tests and obtaining verification results.

9. Claims 36, 37, 46, and 54 are rejected under 35 U.S.C. 103(a) as being unpatentable over Logan, U.S. Patent 6,601,018 in view of Breggin et al., U.S. Patent 6,560,776, and further in view of Donohue, U.S. Patent 6,202,207.

**In regard to claims 36 and 37,** incorporating the rejection of claims 1 and 36 respectively:

*"A software package verification tool as in claim 1 wherein the software package is compliant with the SOLARIS standard."*

*"A software package verification tool as in claim 36 wherein the file list comprises a "pkgmap" file."*

Logan discloses a test suite in an automatic test framework (column 2, lines 19 50) that runs in a number of distributed environments (column 3, line 28 to column 4, line 27), but neither Logan nor Breggin explicitly discloses the SOLARIS, a UNIX-based distributed

environment standard. However, Donohue discloses compatibility with a UNIX based operating system (column 8, lines 31 – 33), at least suggesting a compatibility with the UNIX based SOLARIS standard. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the distributed test environment taught by Logan with the UNIX based distributed system of Donohue, including the porting of a "pkgmap" being a package contents description file in a UNIX-based system, because the Donohue invention provides the means to register components from other networks as taught at column 8, lines 19 – 34 thus extending the integrated test environment of Logan according to a known standard.

**In regard to claim 46,** incorporating the rejection of claim 44:

> *"...wherein the file list that identities the plurality of software components in the software package comprises a Solaris pkgmap file."*

Logan discloses a test suite in an automatic test framework (column 2, lines 19 50) that runs in a number of distributed environments (column 3, line 28 to column 4, line 27), but neither Logan nor Breggin explicitly discloses the SOLARIS, a UNIX-based distributed environment standard. However, Donohue discloses compatibility with a UNIX based operating system (column 8, lines 31 – 33), at least suggesting a compatibility with the UNIX based SOLARIS standard. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the distributed test environment taught by Logan with the UNIX based distributed system of Donohue, including the porting of a "pkgmap" being a package contents description file in a UNIX-based system, because the Donohue invention provides the means to register components from other networks as taught at column 8, lines 19 – 34 thus extending the integrated test environment of Logan according to a known standard.

**In regard to claim 54**, incorporating the rejection of claim 15:

> *"...wherein the file list that identifies the plurality of software components in the software package comprises a Solaris pkgmap file."*

Logan discloses a test suite in an automatic test framework (column 2, lines 19 50) that runs in a number of distributed environments (column 3, line 28 to column 4, line 27), but neither Logan nor Breggin explicitly discloses the SOLARIS, a UNIX-based distributed environment standard. However, Donohue discloses compatibility with a UNIX based operating system (column 8, lines 31 – 33), at least suggesting a compatibility with the UNIX based SOLARIS standard. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the distributed test environment taught by Logan with the UNIX based distributed system of Donohue, including the porting of a "pkgmap" being a package contents description file in a UNIX-based system, because the Donohue invention provides the means to register components from other networks as taught at column 8, lines 19 – 34 thus extending the integrated test environment of Logan according to a known standard.

10. Claims 3 – 7, 9, 10; 16 – 18; 25 – 27, 29, 30, and 32 are rejected under 35 U.S.C. 103(a) as being unpatentable over Logan, U.S. Patent 6,601,018, in view of Breggin et al., U.S. Patent 6,560,776, and further in view of Mastronardi, U.S. Patent 6,346,951.

**In regard to claim 3,** incorporating the rejection of claim 2:

> *"...wherein the framework identifies a priority for each of the test modules."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 4,** incorporating the rejection of claim 3:

> *"...wherein the control module is operable to cause the test modules to be executed sequentially according to the priority identified in the framework for each of the test modules."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 5,** incorporating the rejection of claim 1:

> *"...wherein a mechanism is provided for identifying the at least one test module as being one of active and not active."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active or not. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active or not active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 6, incorporating the rejection of claim 5:**

> *"...wherein the mechanism for identifying the at least one test modules as being one of active and not active is included in the framework."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active or not. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active or not active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 7, incorporating the rejection of claim 5:**

> *"...wherein the mechanism for identifying the at least one test modules as being one of active and not active is included in the control module."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active or not. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active or not active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 9, incorporating the rejection of claim 8:**

> *"...wherein each entry defines a priority for the one of the test modules identified therein."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 10, incorporating the rejection of claim 8:**

*"...wherein the identity of the one of the test modules defines its priority."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) based on the identity of the test in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 16, incorporating the rejection of claim 15:**

*"...wherein a priority for each of the test modules is identified in the framework."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 17, incorporating the rejection of claim 15**

*"...sequentially causing each of the test modules to be executed according to the priority identified for each of the test modules."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 18, incorporating the rejection of claim 15:**

*"...identifying each of the test modules as being one of active and not active."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active or not. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active or not active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 25 incorporating the rejection of claim 24:**

*"...wherein a priority for each of the test modules is identified in the framework."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 26, incorporating the rejection of claim 25:**

*"...sequentially causing the test modules to be executed according to the priority identified for each of the test modules."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 27, incorporating the rejection of claim 24:**

*"...identifying each of the test modules as being one of active and not active."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active or not. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active or not active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 29, incorporating the rejection of claim 28:**

*"... wherein each entry defines a priority of the test module identified thereby."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 30, incorporating the rejection of claim 28:**

*"...wherein the identity of a module defines its priority."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) based on the identity of the test in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 32, incorporating the rejection of claim 31:**

> *"...including repeating steps (b) and (c) to perform a sequence of tests, the order in which the tests are performed being determined by relative priorities assigned to each of the at least test module."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Breggin discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

11. Claim 33 is rejected under 35 U.S.C. 103(a) as being unpatentable over Logan, U.S. Patent 6,601,018 in view of Peter et al., U.S. Patent 6,418,389 (hereinafter referred to as Peter).

**In regard to claim 33:**

> *"a) a first field containing data representing one of a plurality of test modules, each test module being operable to test of at least one parameter of the at least one software component of the package,*
>
> *b) where data representing ones of the test modules may be added to and deleted from the data structure, creating a flexible data structure."*

Logan discloses testing of software packages, but does not disclose a data structure having a field representing one of a plurality of test modules, that may be added or deleted, for testing parameters. However, Peter discloses a data structure having a field representing one of a plurality of test modules, that may be added or deleted, for testing parameters (column 9, line 48 to column 10, line 3; e.g., Figures 4a and 4b. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to combine the software testing function of Logan with the use of a data structure containing a representation of the plurality of test modules which allows the user to select the tests desires as discloses in the abstract of Peters.

12. Claims 34 and 35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Logan, U.S. Patent 6,601,018 in view of Peter et al., U.S. Patent 6,418,389, and further in view of Mastronardi, U.S. Patent 6,346,951.

**In regard to claim 34**, incorporating the rejection of claim 33:

> *"...wherein the data structure further comprises a second data field identifying a priority for each of the test modules represented by the data in the first data field, the priority defining an order of execution of the test modules."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Peter discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active as taught by Mastronardi (column 8, lines 25 – 39).

**In regard to claim 35,** incorporating the rejection of claim 33:

> *"...wherein the data structure further comprises a third data field identifying the one of a plurality of test modules represented by the data in the first data field as being one of active and not active."*

Logan discloses a suite of tests in an automatic test framework, but neither Logan nor Peter discloses a priority for each test module. However, Mastronardi discloses a supervisory module that carries out a test priority sequence (column 8, lines 24 – 31) in order to determine if a certain task is active or not. Therefore, it would have been obvious to one skilled in the art at the time the invention was made to modify the automatic software component test framework of Logan combined with the testing priority Mastronardi, because the testing priority provides a means to determine if certain tasks or modules are active or not active as taught by Mastronardi (column 8, lines 25 – 39).

***Conclusion***

13. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Lawrence Shrader whose telephone number is (571) 272-3734. The examiner can normally be reached on M-F 08:00-16:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571) 272-3719. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Lawrence Shrader
Examiner
Art Unit 2124

4 March 2005

Kakali Chaki

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100